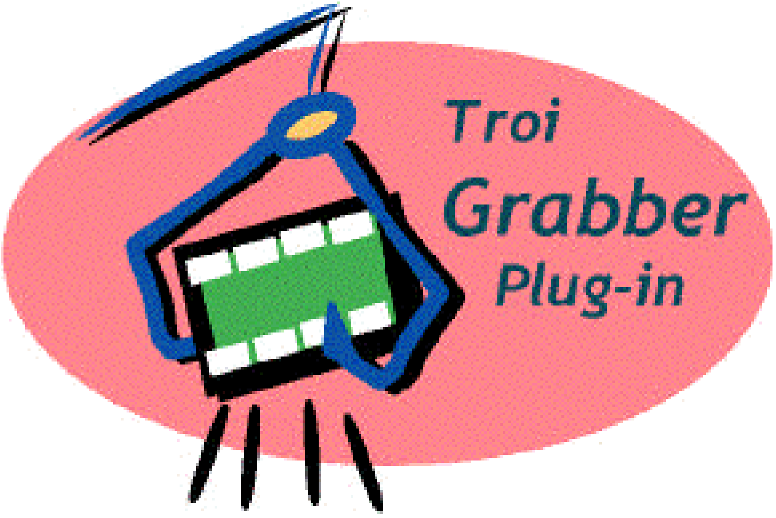


TROI GRABBER PLUG-IN™ 1.2 USER GUIDE

November 1999

User Guide updated 04 November 1999

Troi Automatisering
Vuurlaan 18
2408 NB Alphen a/d Rijn
The Netherlands
Tel: +31 172-426606 Fax: +31-172-470539

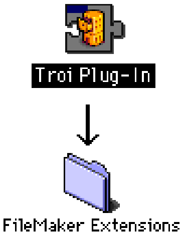
You can also visit the Troi web site at: <http://www.troi.com/> for additional information.

Table of Contents

Installing plug-ins

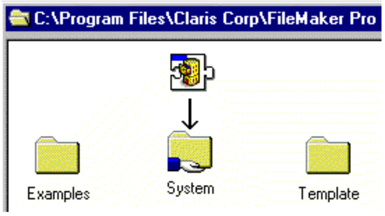
For Macintosh:

- Quit FileMaker Pro.
- Put the file "Troi Grabber Plug-in" from the folder "Macintosh Plug-in" into the "FileMaker Extensions" folder in the FileMaker Pro application folder.
- If you have installed previous versions of this plug-in, you are asked: “An older item named “Troi Grabber Plug-In” already exists in this location. Do you want to replace it with the one you’re moving?’. Press the OK button.
- Start FileMaker Pro. The first time the Troi Grabber Plug-in is used it will display a dialog box, indicating that it is loading and showing the registration status.



For Windows:

- Quit FileMaker Pro.
- Put the file "grabber.fmx" from the directory "Windows" into the "SYSTEM" subdirectory in the FileMaker Pro application directory.
- If you have installed previous versions of this plug-in, you are asked: “This folder already contains a file called 'grabber.fmx'. Would you like to replace the existing file with this one?’. Press the Yes button.
- Start FileMaker Pro. The Troi Grabber Plug-in will display a dialog box, indicating that it is loading and showing the registration status.



TIP You can check which plug-ins you have loaded by going to the plug-in preferences: Choose **Preferences** from the **Edit** menu, and then choose **Plug-ins**.

You can now open the file "GrabExpl.fp3" to see how to use the plug-in's functions. There is also a Function overview available.

IMPORTANT There is a problem in FileMaker Pro 4.0v1. Please make sure that all plug-ins that are in the folder "FileMaker Extensions" are enabled in the preferences. (Under Edit/ Preferences/ Application/ Plug-ins). Make sure all plug-ins have a cross before their name. Remove plug-ins you don't use from the "FileMaker Extensions" folder. This bug is fixed in version 4.0v2, 4.1 and 5.0.

If You Have Problems

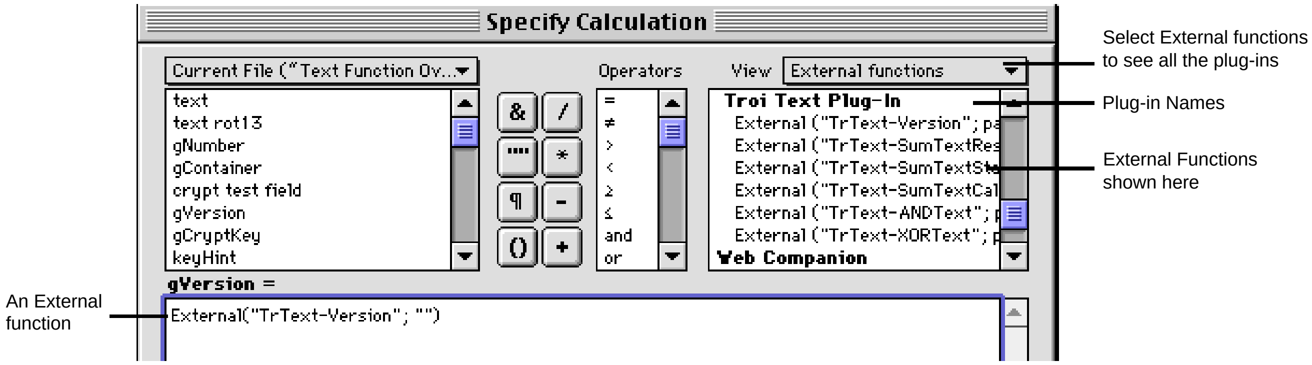
This user manual tries to give you all the information necessary to use this plug-in. So if you have a problem please read this user guide first. If that doesn't help you can get free support by email. Send your questions to **support@troi.com** with a full explanation of the problem. Also give as much relevant information (version of the plug-in, which platform, version of the operating system, version of FileMaker Pro) as possible.

If you find any mistake in this manual or have a suggestion please let us know. We appreciate your feedback!

TIP You can get more information on returned error codes from our OSErrrs database on our website: <html://www.troi.com/software/oserrrs.html>. This free FileMaker database lists all error codes for Windows and Mac OS!

Summary of functions

Plug-ins add new functions to the standard functions that are available in FileMaker Pro. You can see those extra functions for all plug-ins at the top right of the Specify Calculation Box:



Specify Calculation

Current File ("Text Function Ov...

text
text rot13
gNumber
gContainer
crypt test field
gVersion
gCryptKey
keyHint

Operators

View: External functions

Troi Text Plug-In
External ("TrText-Version"; pa
External ("TrText-SumTextRes
External ("TrText-SumTextSt
External ("TrText-SumTextCal
External ("TrText-ANDText"; p
External ("TrText-XORText"; p
Web Companion

gVersion =

External("TrText-Version"; "")

Select External functions to see all the plug-ins

Plug-in Names

External Functions shown here

An External function

IMPORTANT In the United States, commas act as list separators in functions. In other countries semi-colons might be used as list separators. The separator being used depends on the operating system your computer uses, as well as the one used when the file was created. All examples show the functions with comma's.

The Troi Grabber Plug-in adds the following functions:

function name	short description
Grab-Version	check for correct version of the plug-in
Grab-Initialise	initialises the plug-in and checks for available input sources
Grab-StartPreview	(Mac only) begins showing the preview of the image to be captured
Grab-ImageToClip	(Mac only) captures a picture and puts it in the clipboard buffer
Grab-DoSettingsDialog	shows a video settings dialog
Grab-GetSettings	(Mac only) retrieves the current video settings
Grab-SetSettings	(Mac only) sets the video settings to the passed parameter
Grab-Stop	stops the video preview
Grab-AquireToClip	(Win only) aquires a picture and puts it in the clipboard buffer
Grab-SetRotation	(Mac only) rotates the preview.
Grab-DisplayCropRect	(Mac only) displays a cropping rectangle on the preview.

Using external functions

External functions for this plug-in can be used in a script step using a calculation. The functions should not be used in a define field calculation.

Function Reference

Grab-Version

Example usage: External("Grab-Version" , "") will return "Troi Grabber Plug-in 1.2".

IMPORTANT Always use this function to determine if the plug-in is loaded. If the plug-in is not loaded use of external functions may result in data loss, as FileMaker will return an empty field to any external function that is not loaded.

Grab-Initialise

Syntax External("Grab-Initialise" , "")

Use this function before calling any other grabber functions. This function looks if there is a video input source available. If it finds a video source it initialises it and returns the maximum width and height of the video input rectangle, for example "320|240". This returned rectangle can be used in the function to start a video preview: **Grab-StartPreview**.

Parameters

no parameters: leave blank

Returned result

Mac: the width and height of the video input òr an error code. Successful results will be for example:

320|240
100|80

Win: 0 or an error code.

Possible results with error codes are:

$$-2020	quickTime not available
$$-230	no video input device found

Other error codes might be returned.

IMPORTANT In your script always check for an error code in the returned result by checking if the result begins with "$$"

Example Usage

```
Set Field [gWidthHeight, External("Grab-Initialise", "") ]
```

Grab-StartPreview (Mac OS only)

Syntax   External("Grab-StartPreview" , "*left* | *top* | *width* | *height*")

This starts the video preview stream at the specified coordinates and dimensions in the front window. In this rectangle the current video image is shown. see also **Implementing Picture Grabbing.**

IMPORTANT Use the function **Grab-Initialise** first to test if there if an input source available and initialise it.

Parameters

left: the left coordinate (in pixels) of the destination rectangle in the window
top: the top coordinate (in pixels) of the destination rectangle in the window
width: the width (in pixels) of the destination rectangle in the window
height: the height (in pixels) of the destination rectangle in the window

Left and **top** are the coordinates (in pixels) on the window of the upper left point of the destination rectangle. **Width** and **height** give the dimension (in pixels) of this destination rectangle. The desired width and height can be calculated from the results you got from function **Grab-Initialise**. **Grab-Initialise** returns you the maximum width and height of the video rectangle. To get a proportional smaller image divide the width and height by the same factor.

Example Usage

```
    Set Field [gErrorCode,
        External("Grab-StartPreview", "10|100|" & gWidth & "|" & gHeight)") ]
```

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

Possible results with error codes are:

0	no error	
$$-50	parameter error	you supplied the wrong parameters

Other error codes might be returned.

Example

Let's say **Grab-Initialise** returned a width and height of "320 | 240". If you want a quarter of that image to be shown on the window, divide both numbers by 4. Use " 80 | 60" as width and height:

```
    Set Field [gWidth,  "80"]
    Set Field gHeight,  "60"]
    Set Field [gErrorCode,
         External("Grab-StartPreview", "10|100|" & gWidth & "|" & gHeight)") ]
```

These script steps will start the preview at 10 pixels from the left and 100 pixel from the top. The rectangle will be 80x60.

Grab-ImageToClip   (Mac OS only)

Syntax External("Grab-ImageToClip" , "")

This function will grab the current video image and puts it as a picture on the clipboard. Normally you want to put this clipboard picture in a container field. You can use the **paste** script step for this.

NOTE The original contents of the clipboard is lost.

Parameters

no parameters: leave blank

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

Possible results with error codes are:

0	no error	
$$-50	parameter error	there is no video preview showing
$$-108	memFullErr	ran out of memory

Other error codes might be returned.

IMPORTANT In your script always check for an error code in the returned result by checking if the result begins with "$$".

IMPORTANT This function only works when there is already a preview showing. Use the functions **Grab-Initialise** and **Grab-StartPreview** first to get a video preview.

Example

The following script grabs the picture into the clipboard and checks for errors. If there is no error a new record is created. Then the clipboard contents is pasted into the container field **image.**

```
Set Field [gErrorCode, External("Grab-ImageToClip", "")]
If [gErrorCode <> 0]
    Beep
    Show Message [An error occurred....]
    Halt Script
End If
New Record/Request
Paste [Select, image]
Go to Field []
```

Grab-Stop

Syntax External("Grab-Stop" , "")

This function will stop the current video image preview. You can use this function even when you don't have a preview showing.

Parameters

no parameters: leave blank

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.
Possible results with error codes are:

0	no error

In version 1.0 of the plug-in always returns 0.

Example Usage

```
External("Grab-Stop", "")
```

Grab-DoSettingsDialog

Syntax External("Grab-DoSettingsDialog" , "*switches*")

This function shows the video settings dialog. With this dialog you can change settings like brightness, effects etc. Also if you have more than one video input source you can select the wanted source. After the dialog is finished you can use **Grab-GetSettings** to save the selected settings in a text field.

IMPORTANT The video preview must be running first.

Parameters

switches: set to "-video". No other switches are supported at the moment.

Example Usage

```
Set Field [gErrorCode, External("Grab-DoVideoDialog", "-video") ]
```

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

Possible results with error codes are:

0	no error	
$$-50	parameter error	you supplied the wrong parameters

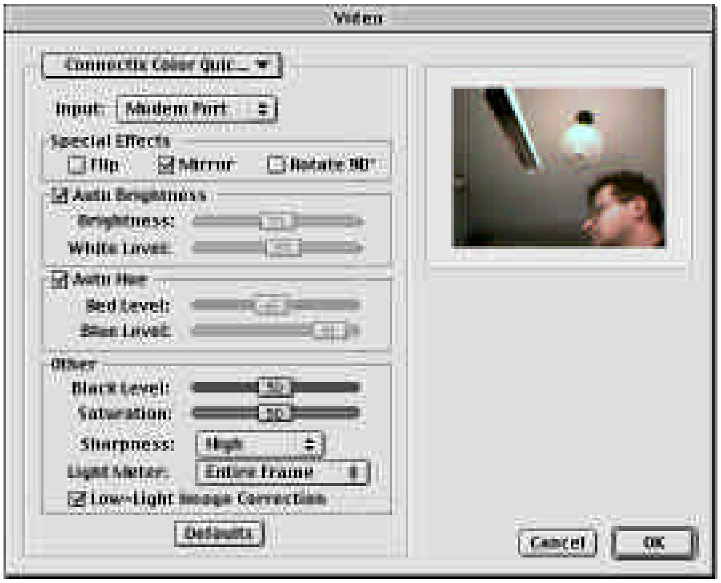
Other error codes might be returned.

Example

This script step will show the video settings dialog:

```
Set Field [gErrorCode, External("Grab-DoVideoDialog", "-video") ]
```

A dialog similar to the one below is shown:



Input: Modem Port

☐ Flip ☑ Mirror ☐ Rotate 90°

☑ Auto Brightness

☑ Auto Hue

Black Level: 50

Saturation: 50

Sharpness: High

Light Meter: Entire Frame

☑ Low-Light Image Correction

Grab-GetSettings (Mac OS only)

Syntax External("Grab-GetSettings" , "")

This returns the current video settings. The format is of this information is not in a human readable form. Store the result for later use in (for example) a global field.

Parameters

no parameters: leave blank

Returned result

If successful this function returns the (encoded) current settings for the video input. If unsuccessful it returns an error code starting with $$ and the error code.

Possible results with error codes are:

$$-50	parameter error	there is no video preview showing
$$-108	memFullErr	ran out of memory

Other error codes might be returned.

IMPORTANT The video preview must be running first. Also check for an error code in the returned result by checking if the result begins with "$$".

IMPORTANT Do not attempt to change these values. If you return these you may crash the application. Only store them so you can reset them later.

Example Usage

```
Set Field [gGrabberSettings, External("Grab-GetSettings", "" )]
```

Grab-SetSettings (Mac OS only)

Syntax External("Grab-SetSettings" , "*videoSettings*")

This sets the video settings to the given video settings.

Parameters

videoSettings: Use only a video setting you got as a result from "Grab-GetSettings"

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

Possible results with error codes are:

0	no error	
$$-50	parameter error	there is no video preview showing
$$-108	memFullErr	ran out of memory

Other error codes might be returned.

REALLY IMPORTANT Do not use changed settings as a parameter. If you do this you may crash FileMaker.

IMPORTANT The video preview must be running first. Also check for an error code in the returned result by checking if the result begins with "$$".

Example Usage

```
Set Field [gErrorCode, External("Grab-SetSettings", gGrabberSettings )]
```

Grab-AquireToClip   (Windows only)

Syntax External("Grab-AquireToClip" , "")

This shows the aquire image dialog box. The live video image preview is shown in a separate window. There you can click on the “Aquire image” button. The image is put on the clipboard.

Parameters

none

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

Possible results with error codes are:

0	no error	
$$-50	parameter error	there is no video preview showing
$$-108	memFullErr	ran out of memory

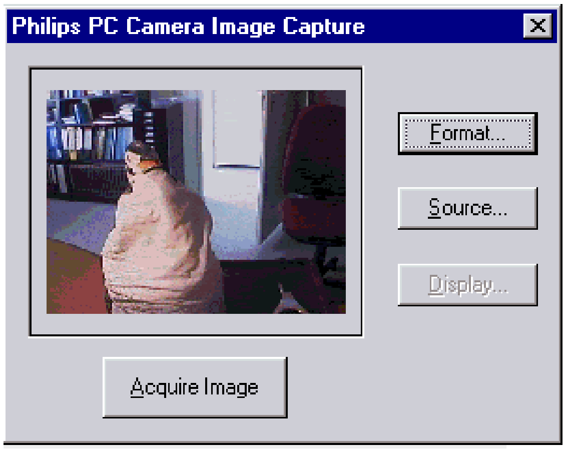
Other error codes might be returned.

IMPORTANT The video preview must be running first. Also check for an error code in the returned result by checking if the result begins with "$$".

Example Usage

```
    Set Field [gErrorCode, External("Grab-AquireToClip", "" )]
```

This script step will show a dialog similar to the one below is shown:



Implementing Picture Grabbing

Implementing video grabbing it not difficult, but the developer of the database must be aware of some limitations to get optimum results.

Plug-in Limitations

The Grabber plug-in has some limitations. Please be aware of the following:

General Limitations

- The plug-in implementation is different for Mac and Windows. The commands for the Windows version are slightly different from the Mac OS version. The Mac version is based on QuickTime, while the Windows version is based on TWAIN drivers.

- **No movie grabbing**: At the moment the grabbing is limited to pictures only. You can not grab a movie.

Limitations for Mac

- **Preview only when window is on top**: The video preview is only shown and updated when FileMaker is the frontmost application. Also the window where the preview is shown must be frontmost. If you switch to a different file or application the video preview is no longer updated.

- **FileMaker is unaware of the video preview**: When you start a video preview, the plug-in shows the image stream in the window. FileMaker is not aware that this is happening. There is no fixed relation to the preview rectangle and the other FileMaker objects on the screen. So when a user hides the status bar the FileMaker objects move to the left but the preview rectangle is still shown on the same coordinates. This is not serious, but looks very strange and may confuse a user. Therefore before starting a video preview, be sure to lock the window down. Use these script steps:

```
Toggle Status Area [Hide, Lock]
Set Zoom Level [100%, Lock]
```

It doesn't matter if the status area is hidden or shown, but make sure to lock it. The same holds true for the Zoom level; choose a fixed zoom level and lock it.

- **Video Preview is not aware of the layout:** Make sure to stop the video preview when the user leaves the layout. The preview rectangle is still shown on the same coordinates if you don't do this. This is also not serious, but looks very strange and may confuse a user.

Also note that when switching to Layout mode the preview can continue to be shown. At the moment we don't have a way to prevent this surprising but harmless behaviour.

- **Video Preview must fit on the window:** Make sure the video preview is completely visible on the screen. Otherwise the capture is only of the visible part of the rectangle.

Limitations for Windows

- TWAIN drivers may be problematic: Working with some TWAIN drivers has proved quite difficult, so the Windows plug-in may not work in your situation. Always test be very careful, save other work first and test this software using a test database. Use at your own risk!

- No Preview in Window: Some TWAIN drivers don't allow to show a preview in the FileMaker window. Therefore this had to be implemented different with the AquireToClip command.

- Aquire Image Window stays black: With some TWAIN drivers the preview image stays black or is not updated after moving the Aquire window.

- Aquire Image Window can be put behind the FileMaker window: The Aquire window may be put behind the FileMaker window..

- Camera needs to be on: Badly written TWAIN drivers may not report the fact that the camera is not connected.This might even crash the system. The plug-in cant detect this. Therefore be sure the camera is connected properly.

- No way to set or get settings.

Steps for creating a picture grabbing database (MacOS)

These are the main steps to create a grabbing database:

1 - create a picture container field and some assisting global fields.
2 - create a new layout with room for the video input rectangle and the picture field on it. You can also modify an existing layout.
3 - create 3 scripts: to start the preview, to capture an image and to stop the preview.
4 - if wanted you can do this in a loop so you grab pictures in sequence.

1- Define Fields

Define the following fields:

image	Container	
gErrorCode	Global	Text
gWidthHeight	Global	Text
gWidth	Global	Text
gHeight	Global	Text

2- Create a Grabber Layout

Create a layout or modify an existing layout. Make room for the video preview image. You can do this by trying out the preview and adjusting the scripts and layout.

3- Create 2 Grabber Scripts

In ScriptMaker define a script "Start Preview". This will get the width and heigth from the video source and starts the video preview:

Define "Start Preview" as follows:

```
Comment [get the maximum dimensions of the video camera]
Set Field [gWidthHeight, External("Grab-Initialise", "")]
If [Left(gWidthHeight , 2)  <>  "$$"]
    Comment [there was no error; split the 2 dimensions]
    Set Field [gWidth, Left(gWidthHeight , Position( gWidthHeight , "|" , 1 , 1 ) - 1)]
    Set Field [gHeight, Right(gWidthHeight , Position( gWidthHeight , "|" , 1 , 1 ) + 1)]
    Comment [start the preview 10 pixels from the left and 100 pixels from the top...]
    Comment [... with the width and height is half of the maximum of the camera]
    Set Field [gErrorCode,
             External("Grab-StartPreview", "10|100|" & gWidth /2  & "|" & gHeight / 2)]
Else
    Comment [An error occurred. put the errorcode into the gErrorCode global.]
    Set Field [gErrorCode, gWidthHeight]
    If [gErrorCode = "$$-9405"]
        Beep
        Show Message [No video component was found on this computer.]
    Else
        Beep
        Show Message [An error occured while initializing the video input.]
    End If
End If
```

In ScriptMaker also define a script "New record and Grab Image To Clipboard" as follows:

```
Set Field [gErrorCode, External("Grab-ImageToClip", "")]
If [gErrorCode <> 0]
     Beep
     Show Message [An error occurred. Make sure you start the preview]
     Halt Script
End If
New Record/Request
Paste [Select, image]
Go to Field []
```

This script will capture the current preview image.

4- Grab Pictures in a loop

If you want to grab images in a loop define a script "Loop: Grab Images into records" as follows:

```
Go to Layout [Image grabbing]
Loop
     Perform Script [Sub-scripts, New record and Grab Image To Clipboard]
     Pause/Resume Script [0:00:10]
     Exit Loop If [gErrorCode <> 0]
End Loop
```

This will grab a picture every 10 seconds.

Steps for creating a picture grabbing database (Windows)

Implementing video grabbing on Windows is slightly different.
These are the main steps to create a grabbing database on windows:

1 - create a picture container field and some assisting global fields.

2 - create a new layout with the picture field on it. You can also modify an existing layout. (On windows the room for the video input rectangle is not needed).

3 - create 2 scripts: to start the image aquiring and to stop the it again.

1- Define Fields

Define the following fields:

image	Container	
gErrorCode	Global	Text

2- Create a Grabber Layout

Create a layout or modify an existing layout. Put the the picture field `image` on it.

3- Create Grabber Script

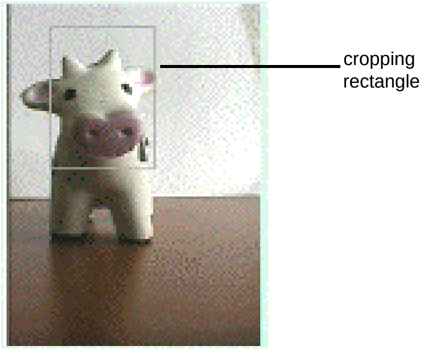
In ScriptMaker define a script "Aquire Image". This will get initialise the video source and show the aquire image preview:

Define "Aquire Image" as follows:

```
Set Field [gErrorCode, External("Grab-Initialise", "")]
If [Left(gErrorCode , 2)  <>  "$$"]
     Comment [there was no error]
     Comment [show the Aquire image window]
     Set Field [gErrorCode, External("Grab-AquireToClip", "")]
End If
Else
If [Left(gErrorCode , 2)  <>  "$$"]
     Comment [there was no error]
     Paste[image]
Else
     Comment [An error occurred.]
     Beep
     Show Message [An error occured while initializing the video input.]
End If
```

Advanced Grabbing (Mac only)

The following features of Troi Grabber plug-in are only available for Mac OS, as they make use of Quicktime. You need version 1.1 or later of the Grabber plug-in for these advanced features. In the picture to the right a sample preview is showing a video preview that is rotated 90° and it shows a cropping rectangle.



Implementing Rotation and Cropping

The Troi Grabber Plug-in for Mac OS can rotate the video preview and crop images. Implementing these features require that you add some special steps to your ScriptMaker scripts.

These are the main steps to take to enable these features:

1 - Initializing the video preview using a "GWorld"
2 - Set the rotation
3 - Initialize the cropping rectangle

NOTE You don't need to implement both step 2 and 3. You can also use rotation or cropping seperately.

What is a GWorld?

The cropping and rotation functions are implemented with the help of a GWorld. A GWorld is a technical term: it is an offscreen part of the computer memory that makes it possible to manipulate the grabbing process.

Initializing the video preview

You don't need to remember what a GWorld is. The main thing to remember is that for Rotation and Cropping you need to initialise the Grabber Plug-in using the **`useGWorld`** tag, like this:

```
    Set Field [gWidthHeight, “External("Grab-Initialise", "useGWorld"))”]
```

TIP A GWorld uses extra memory, so for large video images you might need to add more memory to FileMaker.

Setting the rotation

If you want to rotate the video image generated you use the function `Grab-SetRotation`. Here is a sample script **`Set rotation 90`**:

```
Comment [Set rotation to 90 degrees]
Set Field [gErrorCode, “External("Grab-SetRotation", "90")]
```

Initializing the cropping rectangle

If you want to show a cropping rectangle use the function `Grab-DisplayCropRect`. Here is a sample script **`Show Crop Rectangle`**:

```
Comment [Show a rectangle at the specified coordinates]
Set Field [gErrorCode,
           External("Grab-DisplayCropRect", "100|50|240|320")]
```

The parameters for the `Grab-DisplayCropRect` function are the 4 coordinates of the cropping rectangle in this format: "`left | top | width | height` ". The coordinates are relative to the video source: left and top are the coordinates (in pixels) in the source of the upper left part of the rectangle. width and height give the dimension (in pixels) of the rectangle.

Example script

The following script **`Start Preview Mac RotCrop`** shows the elements together:

```
Comment [Start up the preview of the image to be grabbed]
Set Field [gWidthHeight, External("Grab-Initialise", "useGWorld"]
If [Left(gWidthHeight , 2)  <>  "$$"]
    Set Field [gWidth, Left(gWidthHeight , Position( gWidthHeight , "|" , 1 , 1 ) - 1)]
    Set Field [gHeight, Right(gWidthHeight , Position( gWidthHeight , "|" , 1 , 1 ) + 1)]
    Set Field [gErrorCode,
               External("Grab-StartPreview", "10|100|" & gHeight / 2 & "|" & gWidth /2)]
    Perform Script [Sub-scripts, Set Grabber Settings]
    Perform Script [Sub-scripts, Set rotation 90]
    Perform Script [Sub-scripts, Show Crop Rectangle]
Else
    Comment [An error occurred. put the errorcode into the global.”]
    Set Field [gErrorCode, gWidthHeight]
End If
```

After this script has run the preview should be visible, rotated 90° and with a cropping rectangle.

Getting a cropped image into a container

When you are previewing you can grab the complete image, as well as a part of the image into a container field.

There a 2 methods to crop images:
1 - Crop the image on the clipboard
2 - Get a cropped image from the video preview

Use the first method if is important that the cropped image is exactly the same as (the part of) the larger image.
If you use the second way the cropped image is grabbed freshly from the video source and will be the image that is then showing.

Scenario 1

Storing 2 identical images: 1 full size and 1 cropped

If you want the grabber to grab an image and also store a part of that same image, use this script.

"New record + store 2 identical images: 1 Full + 1 Cropped":

```
Comment [“# 1- get the full size image from the grabber into the clipboard...”]
Set Field [“gErrorCode”, “External("Grab-ImageToClip", "")”]
If [“gErrorCode <> 0”]
    Beep
    Show Message [“An error occurred. Make sure you start the preview before
                  grabbing an image.”]
    Halt Script
End If
New Record/Request
Comment [“... paste it into the 1st container field.”]
Paste [Select, “image”]
Comment [“# 2- crop the image on the clipboard....”]
Perform Script [Sub-scripts, “CropClipboard”]
Comment [“... then  paste it into the 2nd container field.”]
Paste [Select, “image Detail”]
Comment [“Leave the record to save it to disk:”]
Go to Field []
```

This is the script **`CropClipboard`**:

```
Comment [“Crop an image on the clipboard to the requested size.”]
Set Field [“gErrorCode”, “External("Grab-CropClip", "100|50|240|320")”]
If [“gErrorCode <> 0”]
    Beep
    If [“gErrorCode = "$$-4207"”]
        Show Message [“An error occurred. There was no picture in the clipboard.”]
    Else
        Show Message [“An error occurred. There was not enough memory or you have
                      specified incorrect parameters in this script.”]
    End If
    Halt Script
End If
```

TIP: In this example the parameters for Grab-CropClip are hardcoded into the script, but it would be more flexible to store this into global fields.

Scenario 2

Getting full size image and cropped image after each other

If you want the grabber to grab a full size image first then get a cropped image after that use this script.

-New record and Grab 2 Images To Clipboard:

```
     Comment [“# 1- get the full size image from the grabber into the clipboard...”]
     Set Field [“gErrorCode”, “External("Grab-ImageToClip", "")”]
     If [“gErrorCode <> 0”]
          Beep
          Show Message [“An error occurred. Make sure you start the preview before grabbing an
image.”]
          Halt Script
     End If
     New Record/Request
     Comment [“... paste it into the 1st container field.”]
     Paste [Select, “image”]
     Comment [“# 2- get a cropped image from the grabber on the clipboard....”]
     Set Field [“gErrorCode”, “External("Grab-ImageToClip", "cropped")”]
     Comment [“... then  paste it into the 2nd container field.”]
     Paste [Select, “image Detail”]
     Go to Field []
```

Grab-SetRotation (Mac OS only)

Syntax External("Grab-SetRotation" , "*rotationAngle*")

This tells the grabber plug-in how to rotate the video preview.

Parameters

rotationangle: the number of degrees the image must be rotated.

Note: The rotation can only be 0, 90 and -90 degrees.

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

Possible results with error codes are:

0	no error	
$$-50	parameter error	there is no video preview showing, or was not initialised with offscreen grabbing,.
$$-108	memFullErr	ran out of memory
$$-2020	movieToolboxUninitialized	Quicktime is not installed

Other error codes might be returned.

IMPORTANT The video preview must be running first. Also check for an error code in the returned result by checking if the result begins with "$$".

Example Usage

```
Set Field [gErrorCode, External("Grab-SetRotation", "90")]
```

Grab-DisplayCropRect (Mac OS only)

Syntax External("Grab-DisplayCropRect" , "*left* | *top* | *width* | *height*")

This tells the grabber display a cropping rectangle on the preview of the video source.

Parameters

left: the left coordinate (in pixels) of the cropping rectangle in the video source
top: the top coordinate (in pixels) of the cropping rectangle in the video source
width: the width (in pixels) of the cropping rectangle in the video source
height: the height (in pixels) of the cropping rectangle in the video source

These coordinates are relative to the video source. **Left** and **top** are the coordinates (in pixels) in the video source of the upper left point of the cropping rectangle. **Width** and **height** give the dimension (in pixels) of this rectangle.

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.
Possible results with error codes are:

0	no error	
$$-50	parameter error	there is no video preview showing, or was not initialised with offscreen grabbing,.
$$-108	memFullErr	ran out of memory

Other error codes might be returned.

IMPORTANT The video preview must be running first. Also check for an error code in the returned result by checking if the result begins with "$$".

Example Usage

```
Set Field [gErrorCode, External("Grab-DisplayCropRect", "100|50|240|320")]
```